We can understand the principles of security architecture as the general guidelines or fundamentals to be followed to ensure the security of systems and networks, these can be considered good practices and guidelines that are nowadays widely accepted.

Establishing and following these principles then helps us to protect digital assets, prevent cyber-attacks and maintain the integrity, confidentiality and availability of information. They form a foundation on which to build appropriate security strategies, and by adhering to them, organizations strengthen their security posture and reduce risks.

None of this could happen without the main figure in charge of planning and designing security in such contexts: the **security architect**.

1. Defense in depth is a strategy that leverages multiple security measures to protect an organization's assets.

To implement defense in depth in this scenario we could incorporate **multi-factor authentication (MFA),** and add for the device itself (let's assume it's a laptop or smartphone) **mobile device management (MDM)** software that ensures that security policy is enforced on the device, including patches, passwords and other measures. We could also add an **Endpoint Detection and Response (EDR)**, system, a type of next-generation solution that provides detection and response capabilities to safeguard the platform against multiple types of threats, including malware. From a network standpoint, we can include a **firewall** to protect the web server from external threats and selectively allow traffic back to these more sensitive areas. In addition, we can perform **vulnerability scans** on the web and application servers to ensure that they are not susceptible to attack. Finally, when data is required, we apply **encryption** to it, **access controls** to safeguard it.

In this way, we have created a system that relies on **multiple safety mechanisms**, so that if one of them fails, the rest of the system continues to function, and we have no single point of failure.

2. Least Privilege is an information security concept that maintains that a user or entity should only have access to the specific data, resources and applications needed to complete a required task.

The second principle we will explore is the **principle of least privilege**. Essentially, this states that access rights should be granted only to people who need them, are authorized, can justify their access, and only for as long as necessary. For example, if we consider three users, where the first does not have a functional need, so we do not grant access, and the other two users demonstrate a legitimate need, and we provide them with appropriate access. Even for these, we do not grant them access indefinitely, but periodically reevaluate whether access is still justified, and if not, we revoke it.

Another aspect of this principle is system hardening, better known as **hardening**. If we have a web server that runs HTTP by default, which is necessary for web traffic, and has FTP and SSH services enabled for remote connectivity, we should ask ourselves if we need these additional services. If not, we should remove them. **Each service we remove reduces our attack surface.**

In addition, we should remove unnecessary default IDs and change the default credentials of the IDs we keep. For example, if the administrator ID is initially set to "admin", we should replace it with an unfamiliar one.

3. Separation of Duty (SOD) refers to the principle that no user should be given enough privileges to misuse the system on their own.

The third security principle to consider is the concept of **separation of duties which** ensures that there is no centralized point of control. Instead, a scenario is created in which multiple malicious actors would have to agree to compromise the system. An example from the physical world might be the case of two people and a door with two locks; one has the key to the first lock, while the other has the key to the second lock. Separately, they cannot open the door, but if they collaborate, they can. In this scenario, there is no single point of control, which provides separation of duties.

In technology, we can apply this to access approval processes. For example, a user submits a request for access to a database, and an approver reviews it and decides whether to grant or deny it based on a business-justified need. By ensuring that the requestor and the approver are different people, we avoid a single point of control.

Separation of duties requires collaboration to compromise the system, which becomes a challenge when many people have to work together while maintaining secrecy.

4. Secure by Design means that technology products are built in a way that reasonably protects against malicious cyber actors successfully gaining access to devices, data, and connected infrastructure.

The fourth safety principle we will discuss is **safety by design**, which refers to the fact that safety should not be an afterthought. In the same way that when designing a building in a seismic zone that is required to resist movement, you don't build it first and then decide to earthquake-proof it, you build the requirement from the beginning. In technology projects we typically start with the requirements phase, move to design, then to code, install the written code, test it, and finally deploy it to production.

Ideally, we would like to feed this cycle continuously, maintaining a continuous development process, but what we do not want is to wait for the last phases to address security once the solution is already out. Security cannot be an add-on at the end, but must be present throughout the entire process, so we must look at security aspects during the requirements phase, and integrate security into the design, keeping in mind the principles of secure development at every step, ensuring a secure installation, protecting test data, and continuing with testing in production. In this way, security is not something we do at the last minute, but something we incorporate pervasively.

5. Keep it simple, stupid is a design principle that states that designs and/or systems should be as simple as possible

The fifth principle we will explore is known as **K.I.S.S.** for "Keep it Simple, Stupid". In other words, we should not make things more complicated than necessary, as this would make it easier for attackers and more difficult for legitimate users. When we design security measures **we often add complexity** to deter unauthorized access, but sometimes it results in a complex maze for users to navigate, which can discourage intended use, or encourage circumvention of the measures, which is an outcome we want to avoid. If we make it harder to do the right thing than the wrong thing, people will choose the wrong path, so we must make the system secure, but also as simple as possible for legitimate users.

For example, if we set password rules (such as a combination of uppercase and lowercase letters, numbers, special characters, and a minimum length) we also require users to maintain different passwords for each system and to change them periodically. But the set of rules creates a problem that when users find a password that fits the criteria, they use it on multiple systems and write it down somewhere, reducing security, which is exactly what we want to avoid. That's why we say that complexity is the enemy of security, and we want the system to be complex enough to deter attackers, but simple enough for legitimate users.

It should also be noted that beyond the principles to consider, there are some to avoid altogether. The most important is **security by obscurity**, which refers to relying on secret knowledge for the security of a system. What we want instead is an open and observable system. This idea is known as one of **Kerckhoff's principles**, which states that a cryptographic system must remain secure even if all its details, except the key, are known. If security is provided by a black box, we cannot see how it works, and even if the creator claims that it is unbreakable and has tested it, it means that he has not found a way to break it, but that does not mean that it cannot be broken. We apply the same approach to securing networks, applications, and systems.